ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

APLICACIÓN ANDROID PARA GESTIÓN DE UN CLUB DEPORTIVO.

Borja Martinicorena Huguet

Jesús Villadangos Alonso

Pamplona, 16-05-2015

PALABRAS CLAVE

ANDROID[7]: SISTEMA OPERATIVO MÓVIL PROPIEDAD DE GOOGLE.

ECLIPSE[4]: ENTORNO DE DESARROLLO INTEGRADO DE CÓDIGO ABIERTO Y MULTIPLATAFORMA

JAVA[8]: LEN GUAJE DE DESARROLLO

SQLite[9]: SISTEMA DE GESTIÓN DE BASES DE DATOS.

SMARTPHONE: TELÉFON O MÓVIL CON FUNCIONALIDADES AÑADIDAS.

CONTENIDO

1. INTRODUCCIÓN

Una aplicación para la gestión de un Club de Baloncesto es una herramienta que permite almacenar los datos de todas las personas del Club y otros Clubes que conoce, llevar un control de los equipos del club (horarios de entrenamientos, pistas, calendarios), visualizar los señalamientos de partidos de cada jornada, consultar las principales páginas web de su interés, etc. La aplicación está destinada a dispositivos móviles que hagan uso del sistema operativo Android, centrado principalmente en el terminal del Coordinador del Club (Samsung Galaxy S5[11]).

Las acciones que pueden realizarse mediante el uso de esta aplicación consisten en poder llamar, mandar mensaje o e-mail a cualquier persona del club o a otros Clubes, consultar los datos de cada equipo, visitar las principales páginas web de interés del club y consultar los señalamientos de cada jornada.

Por otro lado, el usuario podrá introducir, actualizar o borrar los datos almacenados en la Base de Datos de la aplicación, permitiéndole un uso independiente de la aplicación.

1.1. MOTIVACIÓN

Existen diversos motivos por los que se decidió realizar y desarrollar este proyecto en concreto.

El primero de ellos está relacionado con el auge y popularidad actual de la tecnología a la que está destinada la aplicación, es decir, el éxito de los Smartphone y las posibilidades que ofrecen. De esta forma, se han podido adquirir conocimientos del desarrollo destinado a estas plataformas.

Otro de los motivos consiste en el sistema operativo al que va dirigida la aplicación, en este caso Android, un sistema operativo relativamente nuevo y cada vez más extendido en dispositivos móviles, tanto Smartphone como Tablet, como otros dispositivos con una cuota de mercado cada vez mayor. Además es un sistema operativo de código abierto.

Por último, la posibilidad de desarrollar una aplicación para dispositivos móviles es un ámbito que no se estudia ampliamente a lo largo de la ingeniería por lo que era una buena oportunidad para obtener conocimiento en dicho ámbito y poder desarrollar una aplicación acorde. Además ayudo a un club de Baloncesto del cual soy miembro.

1.2. PLANTEAMIENTO TÉCNICO

En cuanto al planteamiento técnico, la idea principal es realizar una aplicación sencilla, intuitiva y lo más cómoda posible con la idea de no mezclar en la agenda del dispositivo móvil personal datos del Club y propios del usuario. Además, debe ser posible realizar otras funciones relacionadas con el funcionamiento del Club, de forma que se facilite al usuario la tarea de informatizar datos.

En cuanto a los requisitos necesarios para el usuario tan solo es necesario disponer de un dispositivo móvil Android, instalar la aplicación y disponer de conexión a Internet

Para el desarrollo del proyecto se ha empleado un equipo con el sistema operativo Windows 7, trabajando con el entorno de desarrollo Eclipse y el kit de desarrollo de software, o SDK, de Android. Para la base de datos se ha empleado la propia de Android, SQLite, integrada en la propia aplicación.

2. ESPECIFICACIÓN DE REQUISITOS

A continuación se detalla una especificación de requisitos referentes a la aplicación.

2.1. INTRODUCCIÓN

En primer lugar es necesario definir algunas características del proyecto, tales como el ámbito, una visión global y una serie de definiciones que ayudarán a la lectura de la propia especificación.

2.1.1. PROPÓSITO

El propósito del presente apartado es definir los requerimientos que debe tener y cumplir la aplicación desarrollada. Esta especificación de requisitos tiene como objetivo formalizar las funcionalidades y prerrequisitos de forma que haya una base con la que contrastar el desarrollo de la aplicación, así poder realizar el desarrollo de una forma más sencilla y guiada.

2.1.2. ÁMBITO

La aplicación consistirá principalmente en una aplicación destinada a dispositivos móviles Android mediante la cual se realizarán todas y cada una de las distintas acciones posibles, ya sea modificar datos como consultar cualquier información de los miembros del club.

Dentro de las características que implementa la aplicación, el usuario podrá realizar diferentes operaciones como eliminar, actualizar e insertar registros de la Base de Datos, consultar datos de personas, clubs o equipos y visitar páginas web.

La funcionalidad principal podría resumirse en permitir la gestión de una agenda telefónica con una serie de características relacionadas con el Club de baloncesto de una forma cómoda y sencilla mediante un dispositivo móvil de forma que no se dependa de un ordenador para el acceso o la modificación de dicha información.

2.1.3. DEFINICIONES, ACRÓNIMOS Y ABREVIATURAS

- **Android:** Es el sistema operativo de Google destinado a dispositivos móviles, Smartphone, Tablet, etc. Lanzado en septiembre del año 2008 ha revolucionado el mundo de los smartphones debido a la cantidad de dispositivos que hacen uso de Android. Las últimas cifras apuntan a más de 225 mil millones de dispositivos con Android y dispone ya de más de 1.000.000 aplicaciones disponibles.

- **JAVA**: Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos 10 millones de usuarios reportados.

- **SQLite:** SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (~275 kiB) biblioteca escrita en C. SQLite es un proyecto de dominio público creado por D. Richard Hipp.

- **Smartphone:** Es el término empleado para denominar a un teléfono móvil con más funcionalidades que un teléfono móvil común.

2.1.4. VISIÓN GLOBAL

En el resto del documento se tratará con detalle las características del proyecto a desarrollar, mostrando los objetivos y características marcadas que la aplicación final debe cumplir.

2.2. DESCRIPCIÓN GENERAL

A continuación se detallan una serie de apartados que describen el objetivo del producto, así como su funcionalidad y requisitos.

2.2.1. PERSPECTIVA DEL PRODUCTO

Para el desarrollo de la aplicación Android se emplea el SDK Android utilizando el lenguaje de programación Java. Dentro de la aplicación se utiliza una base de datos SQLite.

En cuanto a la aplicación Android en sí, el objetivo es desarrollar una aplicación de fácil acceso y aprendizaje, mostrando todas las opciones disponibles de la forma más sencilla. La presentación de la interfaz debe mostrarse sencilla e intuitiva.

Para el desarrollo de la aplicación se hará uso del sistema operativo Microsoft Windows 7. Además se empleará el entorno de desarrollo integrado Eclipse con el SDK Android.

2.2.2. FUNCIONES DEL PRODUCTO

Las funciones principales que la aplicación debe permitir son las siguientes:

1) Gestión de Personas
   - Añadir Persona
   - Eliminar Persona
   - Actualizar Persona
   - Consultar Persona
   - Realizar llamada
   - Mandar Email
   - Mandar mensaje

2) Gestión de clubes
   - Añadir Club
   - Eliminar Club
   - Actualizar Club
   - Consultar Club
   - Realizar llamada
   - Mandar Mensaje

3) Gestión de equipos
   - Añadir Equipo
   - Eliminar Equipo
   - Actualizar Equipo
   - Consultar Equipo

4) Consultar señalamientos

5) Visitar páginas web

6) Consultar registro de llamadas
   - Borrar historial de llamadas

2.2.3. CARACTERÍSTICAS DEL USUARIO

La aplicación debe poder ser manejada por un único usuario, el coordinador del club. Este usuario debe ser capaz de realizar todas las operaciones y funciones comentadas en el apartado anterior. Un detalle importante a tener en cuenta es que la aplicación esta destina a un único usuario, el cual tiene permiso para acceder a los datos privados de todas las personas del club.

2.2.4. RESTRICCIONES GENERALES

En cuanto a las restricciones que tiene el proyecto se puede mencionar la necesidad de trabajar con una base de datos con una estructura preestablecida. En caso de modificarse dicha estructura o eliminarse parcialmente, la aplicación funcionaría de manera incorrecta y no deseada o incluso podría dejar de ser funcional. En caso de tener que modificar la estructura de la base de datos, sería necesario realizar correcciones en la aplicación y/o en las funciones utilizadas para el acceso a la base de datos.

A la hora de emplear la aplicación hay que tener en cuenta que es necesario disponer de acceso a internet para ciertas funcionalidades de la aplicación. Esto puede realizarse mediante conexión WiFi a la red del centro en el que se use la aplicación o mediante conexiones móviles 3G, 4G, etc.

Como la aplicación es para un solo usuario y este tiene permiso para acceder a todos los datos, no he considerado añadir restricciones de seguridad adicionales a las que el propio móvil pueda tener.

2.2.5. SUPUESTOS Y DEPENDENCIAS

En cuanto a las dependencias cabe mencionar que la aplicación funciona únicamente en dispositivos móviles Android. La interfaz se ha diseñado específicamente para el terminal del coordinador, en este caso un Samsung Galaxy S5.

2.3. REQUISITOS ESPECIFICOS

En el siguiente apartado se detallan los requisitos de la aplicación, tanto a nivel funcional como de interfaz, eficiencia y diseño.

2.3.1. REQUISITOS FUNCIONALES

- Buscar persona: el usuario seleccionará el nombre y apellidos a buscar. A continuación se mostrará la lista de las personas que coincidan con esos criterios de búsqueda. Al seleccionar una persona podrá consultar sus datos, llamar, mandar un email o escribir un mensaje.

- Consultar equipos: el usuario accederá a una pantalla donde se listaran los equipos del club. Al seleccionar uno de ellos se mostrara los jugadores y una foto del equipo, y podrá consultar sus horarios de entrenamientos y el calendario de partidos de este equipo.

- Consultar clubes: el usuario seleccionará un club de la lista de clubes y a continuación podrá seleccionar entre dos opciones, llamar o mandar mensaje.

- Consultar señalamientos: el usuario accederá a una pantalla que le mostrara los partidos que disputa cada equipo. Los equipos que juegan como visitantes aparecerán con la fila de color gris y si pulsan sobre la fila podrán buscar en el mapa la localización del polideportivo. Ademas podrá consultar los árbitros designados por la F.N.B.

- Consultar registro de llamadas: el usuario podrá ver un listado de llamadas que no ha podido atender. Podrá devolver la llamada desde esta pantalla. También podrá borrar el historial de llamadas.

- Consultar favoritos: el usuario podrá ver las personas con las que interactúa más frecuentemente, previamente seleccionadas por él y podrá realizar las funciones de llamada, mensaje o email.

- Visitar páginas web: El usuario accederá a una pantalla con los iconos de las principales páginas web de interés del club para así facilitarle el acceso desde la aplicación.
- Gestionar la base de datos: el usuario podrá eliminar, actualizar o crear cualquier persona, club o equipo que quiera.

2.3.2. REQUISITOS DE INTERFAZ

Con respecto a los requisitos de interfaz podemos diferenciar tres tipos: interfaz de usuario, interfaz software e interfaz hardware.

En cuanto a la interfaz de usuario, el principal objetivo al desarrollarla es conseguir una interfaz sencilla de manejar. Puesto que la aplicación es una aplicación destinada a móviles y con diversas funciones no existe un patrón básico que compartan las distintas interfaces. Aun así, debido al carácter de la aplicación y su funcionalidad orientada al uso de agenda y por tanto la necesidad del uso de listas, la interfaz más común es la empleada por las funciones de buscar personas y consultar equipos, las cuales consisten en una lista de personas y equipos, con la posibilidad de interactuar con los datos introducidos previamente en la base de datos. Por otro lado, a la hora de crear una persona, equipo o club, la interfaz consiste en diversos campos a rellenar. Por último, para la funcionalidad de visitar páginas web, se dispone de una interfaz con iconos de cada página, mediante los cuales es posible acceder a las distintas páginas.

En referencia a la interfaz software, el proyecto se desarrolla empleando el sistema operativo Microsoft Windows 7, empleando el entorno de desarrollo Eclipse. Java es el lenguaje principal puesto que Android hace uso de dicho lenguaje. Por último para el sistema de gestión de base de datos se emplea SQLite, sistema propio de la aplicación.

Como requisito o interfaz hardware es importante mencionar la necesidad de emplear un dispositivo móvil con el sistema operativo Android y conexión a Internet.

2.3.3. REQUISITOS DE EFICIENCIA

El uso de la aplicación se realiza de forma individual y única, usando el usuario su dispositivo.

Por otro lado, puesto que la aplicación se ejecuta en un dispositivo móvil la eficiencia de la propia aplicación dependerá de los requisitos de éste, a pesar de esto, cualquier terminal capaz de disponer del sistema operativo Android debería ser capaz de ejecutar la aplicación de forma eficiente.

2.3.4. RESTRICCIONES DE DISEÑO

A la hora de desarrollar una aplicación móvil para dispositivos móviles no existen unos estándares básicos que seguir.

2.3.5. ATRIBUTOS

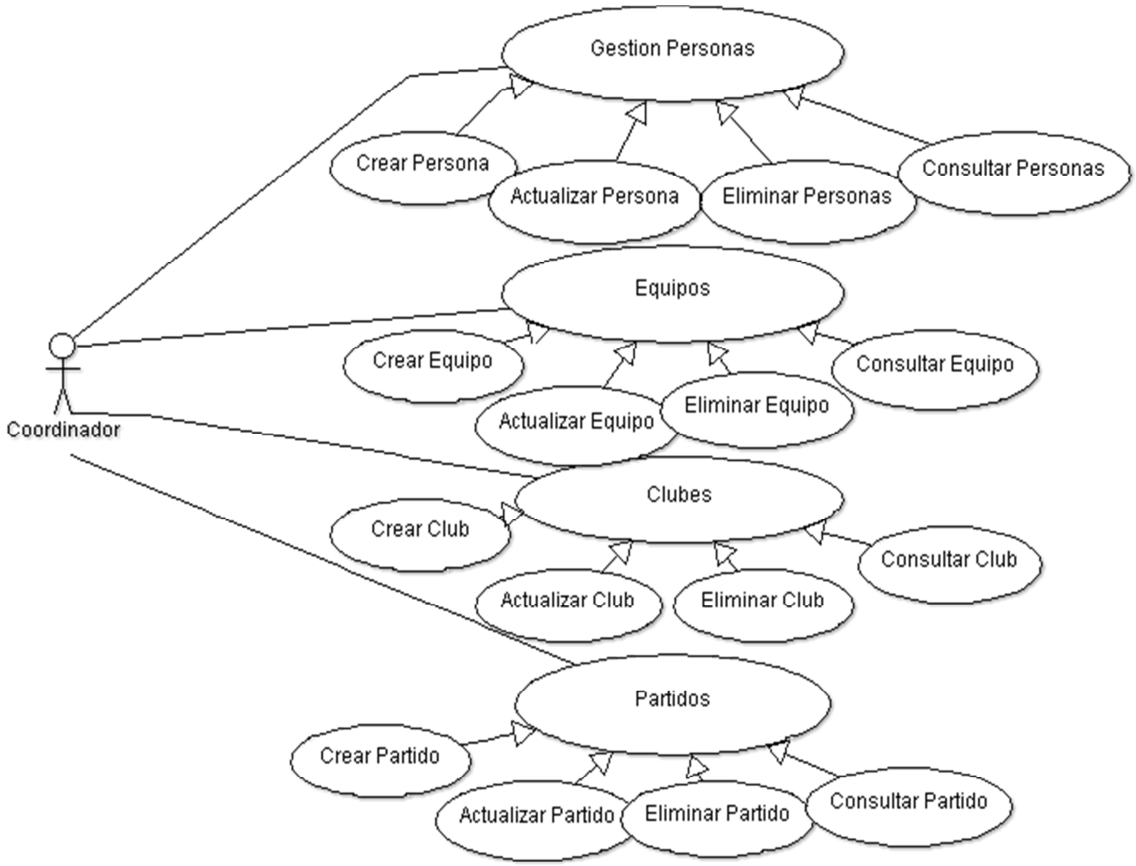
La aplicación creada no debería necesitar mantenimiento puesto que funciona mediante los datos almacenados en la base de datos, así pues, el mantenimiento de los datos recae sobre el usuario ya que será el encargado de actualizarlos o modificarlos mediante el uso de la aplicación.

3. ANÁLISIS

A continuación se detallan los documentos y datos analizados y utilizados en el desarrollo de la aplicación.

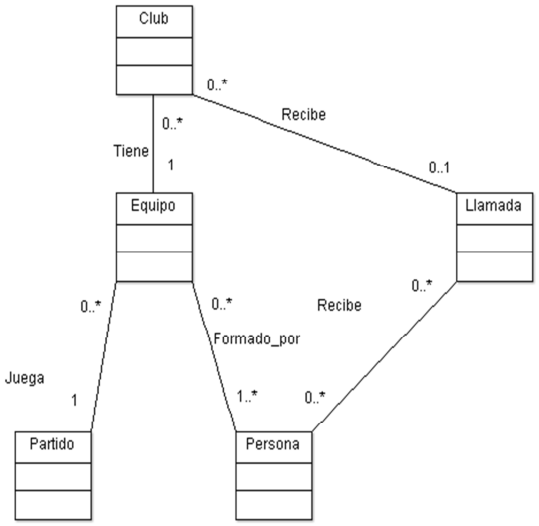
3.1. CASOS DE USO

En la etapa de análisis se hace uso de la metodología UML[12]. El primer punto es la realización de los casos de uso. En este caso, como característica o rasgo, podemos destacar que al existir únicamente un usuario todas las acciones los lleva a cabo el mismo, por lo tanto solo es necesario un diagrama de caso de uso para especificar toda la funcionalidad de la aplicación.



3.2. DIAGRAMAS DE CLASES

El siguiente diagrama corresponde al diagrama de clase. La realización de este diagrama ayuda al entendimiento del sistema. Del diagrama de casos de uso se puede extraer una serie de clases necesarias y que ayudan al desarrollo de la aplicación, específicamente en la capa de lógica con el desarrollo de las clases necesarias.



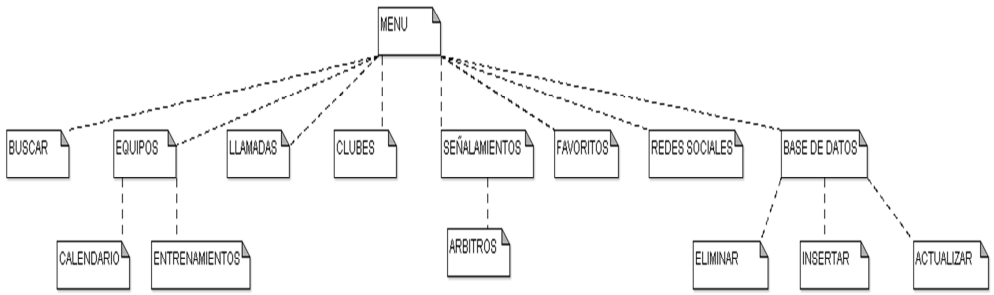
4. DISEÑO

Puesto que se hace uso de un lenguaje orientado a objetos, se emplea el patrón modelo-vista-controlador, dividiendo la aplicación en tres niveles, el nivel de datos correspondiente al modelo, el nivel de interfaz correspondiente a la vista y la lógica de la aplicación correspondiente al controlador.

4.1. VISTA

La vista corresponde con la interfaz de usuario que se muestra. El punto principal de la interfaz es la simplicidad de ésta, puesto que la intención es que la aplicación sea rápida, intuitiva y fácil de usar. Por otra parte, gracias a que Android emplea el patrón anteriormente comentado, modelo-vista-controlador, en caso de querer realizar cambios en la interfaz, tan solo tendremos que modificar la propia interfaz, sin alterar la lógica o funcionamiento de la aplicación.

En el siguiente diagrama se muestra de forma sencilla la navegación por las distintas funcionalidades, de forma que sea sencillo visualizar cómo acceder a la información y la estructura interna de la aplicación.



4.2. CONTROLADOR

En el controlador encontramos el código necesario para que la aplicación realice las funciones que se esperan de ella, es decir el código que se ejecuta cuando el usuario interacciona con la aplicación y ésta debe responder a la interacción. Se ha utilizado orientación a objetos para realizar la lógica de la aplicación, y como ya se ha comentado en el apartado correspondiente a la vista, debido a que Android hace uso del patrón modelo-vista-controlador, podemos modificar detalles del controlador sin que esto afecte al resultado mostrado.

4.3. MODELO

El modelo consiste en el comportamiento e información del dominio de la aplicación, así como del acceso a la base de datos y la propia base de datos.

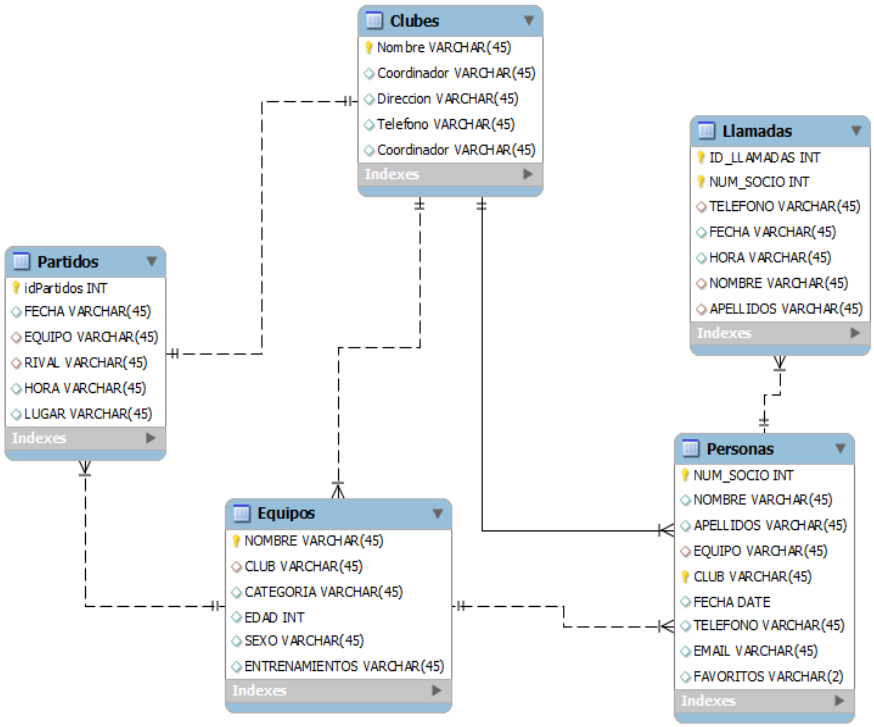
En cuanto al comportamiento e información del dominio destaca la colección de distintas clases empleadas para el desarrollo de la aplicación y vienen establecidas por el diagrama de clases anteriormente mostrado. Esto permite encapsular la información obtenida de forma que pueda manejarse de forma cómoda y sencilla mediante el apartado controlador.

En cuanto a la base de datos, es donde se almacena la información. Así pues, esta parte hace uso de un sistema de gestión de base de datos, el cual en este caso es SQLite.

En la base de datos almacenamos la información relacionada con las personas, equipos, clubes, etc. De esta forma, podemos acceder a ella mediante la aplicación, introducir nuevos datos, eliminar datos o modificarlos. Para poder realizar este mantenimiento de la información se ha habilitado una opción dentro del menú principal donde el usuario podrá acceder a estas opciones de manera sencilla mediante una interfaz intuitiva.

A la hora de desarrollar la base de datos se realizó un diagrama entidad relación donde se plasmaron las entidades más relevantes. Este diagrama fue evolucionando y modificándose, añadiendo nuevas entidades para añadir información, mejorar la manera de acceder a los datos y mejorar su facilidad de uso.

En el siguiente diagrama se puede observar el diagrama entidad-relación empleado para modelar la base de datos.



5. IMPLEMENTACIÓN E INTEGRACIÓN

En este apartado se detallan las distintas tecnologías, herramientas, así como detalles de la implementación e integración.

5.1. TECNOLOGÍAS

La aplicación se ha desarrollado empleando las siguientes tecnologías:

Microsoft Windows 7 como sistema operativo sobre el que se ha desarrollado el proyecto. Lanzado al mercado en octubre del año 2009 y en la actualidad, tras más de 6 años, el sistema operativo Windows 7 aún sigue en uso, aunque decrece el número de usuarios del sistema operativo en favor de la última versión lanzada por Microsoft, Windows 8, a la venta desde 2012

Android de la empresa Google como sistema operativo bajo el cual la aplicación se ejecuta. En la actualidad, junto con iOS, Android es el sistema operativo móvil más utilizado y con un mayor número de aplicaciones disponibles y desarrolladores, lo que lo convierte en un sistema operativo con un amplio número de usuarios.

Samsung Galaxy S5 como el terminal al que va destinado la aplicación. Es el nombre común de los teléfonos de alta gama diseñados por Samsung. Fue presentado oficialmente en Abril de 2014. El sistema operativo con el que se lanzo fue Android 4.4 KitKat, actualizable en la actualidad a Android 5.0 Lollipop.

5.2. HERRAMIENTAS

En cuanto a las herramientas usadas para el desarrollo de la aplicación son las siguientes:

Para el desarrollo del código fuente de la aplicación se ha empleado Eclipse como entorno de desarrollo integrado. Además, se ha usado el SDK de Android, el cual funciona con Eclipse a la perfección, gracias a la capacidad del entorno permitir el uso de plugins.

5.3. DETALLES DE LA IMPLEMENTACIÓN

A la hora de desarrollar la aplicación se ha realizado especial hincapié en la funcionalidad de la misma. Así pues, la aplicación se desarrolló con el objetivo de que el usuario final pudiera realizar cualquier función sin necesidad de mantenimiento. Por lo tanto se le permite hacer uso de la Base de Datos, pudiendo eliminar, actualizar o insertar registros.

El aspecto de la interfaz ha quedado en un segundo plano, puesto que el usuario no requería una interfaz muy bonita sino que le fuese útil. A pesar de ello he tratado de realizar una interfaz lo más intuitiva posible con los recursos de los que disponía.

La aplicación está destinada para un dispositivo móvil concreto, por lo que la interfaz está diseñada para ajustarse a los parámetros de ese Smartphone en concreto (Samsung Galaxy S5).

CONSTRUCCIÓN DE INTERFACES

Dentro del código reseñable cabe mencionar entre otras cosas la construcción de interfaces en Eclipse mediante el SDK de Android. Para analizarlo se tomará de ejemplo la interfaz realizada para el menú de inicio.

A continuación se muestra el código fragmentado por partes.

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/black"
    >
    <TableRow
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="80dp"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        >
          <Button
                android:id="@+id/Buscar_Main"
                style="@style/main_button"
                android:text="@string/buscar"
                android:layout_marginRight="10dp"
          >
          </Button>
          <Button
                android:id="@+id/Equipos_Main"
                android:text="@string/equipos"
                style="@style/main_button"
          >
          </Button>
    </TableRow>
```

La mayoría de las pantallas están diseñadas utilizando tablas, ya que era la manera más sencilla de colocar los botones siguiendo un orden. Vemos que todos los elementos de la pantalla van a estar contenidos en un <TableLayout> y dentro de este contenedor definiré lo que va en cada fila de la tabla dentro de la etiqueta <TableRow>.

Algunos de los atributos de la tabla que definimos son '***android: orientation="vertical"***', para definir la posición de la pantalla, '***android: layout_width="fill_parent"***', en este caso para definir que la tabla tomara todo el ancho de la pantalla. Cabe destacar las opciones que hay en este atributo, por un lado esta fill_parent o match_parent a partir de la API 8 y por otro lado esta wrap_content. En este último caso el elemento definido con esta característica se adaptara al tamaño de la vista.

Uno de los elementos que añadiremos a nuestra tabla dentro de la fila en este caso son los botones. Creamos dos botones en cada fila de tal manera que nos queda una tabla con 2 columnas. Todos los componentes disponen de un atributo *android_id* que sirve para identificarlos. Anteponiendo *@+id/* al id del botón estamos indicando que el id debe añadirse al fichero R.java, un fichero generado de forma automática que permite acceder a los elementos de la interfaz en el código. Además le asignamos un texto que estará dentro del archivo de string, de tal manera que no metemos ningún texto directamente. Por ultimo cabe destacar que para los botones les he creado un estilo sencillo, con un degradado de 2 colores.

```
<?xml version="1.0" encoding="utf-8"?>
<selector
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item>
        <shape>
             <corners
                android:radius="2dp" />
             <gradient
                android:endColor="@color/white"
                android:startColor="@color/Blue"
                android:angle="270" />
            <stroke
                android:width="3dp"
                android:color="@color/AliceBlue"
                 />
            <padding
                android:left="10dp"
                android:top="10dp"
                android:right="10dp"
                android:bottom="10dp" />
            <size
                android:width="80dp"
                android:height="50dp"/>
        </shape>
    </item>
                    </selector>
```

Otro atributo destacable es *android:layout_marginLeft* puesto que se puede observar que el valor no está en píxeles o medidas similares, sino en *dip*, que no es otra cosa que pixeles independientes de la densidad, de forma que se mantiene la proporción al emplear distintas pantallas.

ACCESO A LA BASE DE DATOS

Como ya he mencionado anteriormente, la BBDD es interna, por lo que el acceso es sencillo. Cabe destacar la clase con la que la creo:

```
public class CreateDB extends Activity{
    private final String BD_NOMBRE = "MUTILBASKET";

    /** Se llama a este método cuando accedemos a la Actividad por primera vez. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);


       SQLiteDatabase myDB = null;

       /* Abrimos la base de datos.
        * Si no existía previamente se creará automáticamente. */
       myDB = this.openOrCreateDatabase(BD_NOMBRE, 1, null);

       /* Creamos la tabla de usuarios en la base de datos.
```

```
       * En caso de que existiera previamente no da error
('IF NOT EXISTS'). */


      myDB.execSQL("CREATE TABLE IF NOT EXISTS Clubes
 (nombre VARCHAR PRIMARY KEY,coordinador VARCHAR,telefono
VARCHAR, direccion VARCHAR,color VARCHAR);");
      myDB.execSQL("CREATE TABLE IF NOT EXISTS Equipos
(nombre VARCHAR PRIMARY KEY , club VARCHAR, categoria
VARCHAR, Edad INTEGER, Sexo VARCHAR, entrenamientos TEXT,
FOREIGN KEY(club) REFERENCES Clubes(nombre), CHECK (sexo IN
('Masculino','Femenino')));");
      myDB.execSQL("CREATE TABLE IF NOT EXISTS Personas
(num_socio INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
nombre VARCHAR, apellidos VARCHAR, equipo VARCHAR, club
VARCHAR, fecha DATE, telefono VARCHAR, email VARCHAR,
favoritos VARCHAR, FOREIGN KEY(equipo) REFERENCES
Equipos(nombre), FOREIGN KEY(club) REFERENCES Clubes
(nombre) );");
      myDB.execSQL("CREATE TABLE IF NOT EXISTS Partidos
(id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, fecha
DATE, equipo VARCHAR,rival VARCHAR, hora VARCHAR, lugar
 VARCHAR,CHECK (lugar IN ('Local', 'Visitante')),
FOREIGN KEY(equipo) REFERENCES Equipos(nombre),FOREIGN KEY(rival)
REFERENCES Clubes(nombre));");
      myDB.execSQL("CREATE TABLE IF NOT EXISTS Favoritos
(nombre VARCHAR ,apellidos VARCHAR, telefono VARCHAR, PRIMARY
KEY(nombre,apellidos,telefono),FOREIGN KEY(nombre,apellidos
,telefono) REFERENCES Personas(nombre,apellidos,telefono));");
      myDB.execSQL("CREATE TABLE IF NOT EXISTS Llamadas
(id INTEGER PRIMARY KEY AUTOINCREMENT,telefono VARCHAR,fecha
VARCHAR, hora VARCHAR, nombre VARCHAR, apellidos VARCHAR,FOREIGN
KEY(telefono,nombre,apellidos) REFERENCES Personas
(telefono,nombre,apellidos), FOREIGN KEY(telefono,nombre)
REFERENCES Clubes(telefono,nombre));");
      /* Comprobamos si se ha creado la base de datos */
      if (myDB != null){
        myDB.close();
            finish();
      }
      finish();

   }
                          }
```

Esta clase se ejecuta para crear la Base de Datos la primera vez que ejecuto la Aplicación, las siguientes veces puesto que ya está creada solo serán comprobaciones, aunque se ejecutara igualmente.

```
public void InsertarDatos() {


//Borramos las tablas antes de las insercciones

//Borrar Tablas Equipos
      myDB.execSQL("DROP TABLE IF EXISTS Clubes");
```

```
      **Aqui va el resto de DROP
 //Creacion de Tablas
myDB.execSQL("CREATE TABLE IF NOT EXISTS Clubes
(nombre VARCHAR PRIMARY KEY,coordinador VARCHAR,teléfono
 VARCHAR, direccion VARCHAR,color VARCHAR);");

**Aquí va el resto de CREATE TABLE

//Creacion de equipos

myDB.execSQL("INSERT INTO Equipos (nombre, club, categoría
, edad, sexo, entrenamientos) VALUES ('Senior Masculino',
'C.D. MUTILBASKET', 'Senior', 18,'Masculino',
'Martes(PAB1) 20:30 a 22:00, Jueves(EXT3/EXT4) 20:45 a
22:00');");

 **Aquí va el resto de insert
```

Solo he mostrado un fragmento de esta clase ya que contiene datos privados que no puedo mostrar. La clase se encarga de poblar la Base de Datos la primera vez que se hace uso de la aplicación y en caso de que el usuario lo necesite, restaurar la Base de Datos a su estado inicial. Esto es tarea del usuario, las tablas no se rellenan automáticamente sino que es el usuario quien debe ejecutar esta clase.

También he creado clases para eliminar, actualizar e insertar registros de cada una de las tablas en la base de datos.

Por ultimo pongo un fragmento de una clase con sus métodos en los que trato el acceso a la Base de Datos para las diferentes consultas que realizo en el código para traer registros y tratarlos.

```
public class Registros_SQLite{
      SQLiteDatabase myDB = null;

      public Registros_SQLite(SQLiteDatabase myDB){
            this.myDB=myDB;

      }


    public ArrayList<Equipo> recupera_registros_equipos(){


      ArrayList<Equipo> Equipos = new ArrayList<Equipo>();
        Cursor c = myDB.rawQuery("SELECT * FROM Equipos", null);
```

```
        //Nos aseguramos de que existe al menos un registro
        if (c.moveToFirst()) {
             //Recorremos el cursor hasta que no haya más registros
             do {
                  String Nombre = c.getString(0);
                  String Club = c.getString(1);
                  String Categoria = c.getString(2);
                  String Edad = c.getString(3);
                  String Sexo = c.getString(4);
                  String Entrenamiento = c.getString(5);
                  Equipo Equ = new
 Equipo(Nombre,Club,Categoria,Edad,Sexo,Entrenamiento);
                  Equipos.add(Equ);

             } while(c.moveToNext());
        }
        myDB.close();
             return Equipos;
                        }
```

DESARROLLO DEL CÓDIGO DE LA APLICACIÓN ANDROID

Al hablar de detalles de implementación es necesario mencionar algunas particularidades de Android como las mostradas en el siguiente código:

```
public class Main extends Activity {
 /** Called when the activity is first created. */
 private static final int DIALOGO_BUSQUEDA = 1;
  @Override
 public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);

   final Button buscar = (Button) findViewById(R.id.Buscar_Main);
   final Button equipos = (Button) findViewById(R.id.Equipos_Main);
   final Button clubes = (Button) findViewById(R.id.Clubes_Main);
   final Button senalamientos = (Button) findViewById(R.id.Notas_Main);
   final Button llamadas = (Button) findViewById(R.id.RegistroLlamadas_Main);
   final Button favoritos = (Button) findViewById(R.id.Favoritos_Main);
   final Button redessociales = (Button) findViewById(R.id.RedesSociales_Main);
   final Button basedatos = (Button) findViewById(R.id.BaseDatos_Main);


   buscar.setOnClickListener(new View.OnClickListener() {
         @SuppressWarnings("deprecation")
              @Override
         public void onClick(View v) {

              showDialog(DIALOGO_BUSQUEDA);
         }
   });
 equipos.setOnClickListener(new View.OnClickListener() {
     @Override
      public void onClick(View v) {


   Intent intent = new Intent().setClass(getBaseContext(), Equipos.class);
   startActivity(intent);
         } });
```

El código mostrado corresponde al evento `OnClickListener` disparado cuando se pulsa sobre una botón. Por otra parte en este código podemos ver como se crean nuevas vistas para mostrarlas mediante los *intent* al final del código.

Para añadir una nueva vista, llamadas *activities* en Android, tan solo debemos crear un objeto del tipo *intent* con el fichero *.class* que queramos utilizar y llamar al método *startActivity()* con el objeto de tipo *Intent* como parámetro. Cabe recordar que es necesario declarar todos y cada uno de los archivos *.class* que se vayan a utilizar en el fichero *AndroidManifest.xml,* que se emplea para añadir permisos tanto a nivel de funcionalidades, como por ejemplo permitir la conexión a Internet, como a nivel de los *activities* que se van a ejecutar. En caso contrario la aplicación dará error al intentar usar un fichero *.class* que no se ha declarado en el fichero *AndroidManifest.xml. Ademas en el archivo AndroidManifest.xml es necesario declarar los permisos que se le conceden a la aplicación. Aquí un ejemplo de los permisos que necesita la aplicación:*

```
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.WRITE_SMS"/>
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Los datos añadidos, por ejemplo, *intent.putExtra* se emplean para pasar objetos entre *activities*, de forma que los objetos se pasan de la *activity* actual a la nueva que se está creando, de forma que no se tiene que volver a crear el objeto, acceder a su información en la base de datos, etc.

También comentar la manera de mostrar ventanas emergentes. Esto se realiza mediante el siguiente código:

```
private Dialog crearDialogoRestaurar()
 {
     AlertDialog.Builder builder = new AlertDialog.Builder(this);

     builder.setTitle("Confirmacion");
     builder.setMessage("¿Confirma que quiere RESTAURAR la base de datos?");
     builder.setPositiveButton("Aceptar", new OnClickListener() {
     public void onClick(DialogInterface dialog, int which) {
       pDialog = new ProgressDialog(BaseDatos.this);
              pDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
              pDialog.setMessage("Procesando...");
              pDialog.setCancelable(true);
              pDialog.setMax(100);

              tarea2 = new MiTareaAsincronaDialog();
              tarea2.execute();
     }
     });
     builder.setNegativeButton("Cancelar", new OnClickListener() {
     public void onClick(DialogInterface dialog, int which) {
         dialog.cancel();
     }
     });

     return builder.create();
 }
```

En este caso sencillo se pide una confirmación por parte del usuario a la hora de restaurar la Base de Datos. El mensaje contiene 2 botones, uno que es 'Aceptar' para que llame a la clase que ejecuta la restauración de la Base de Datos y otro de 'Cancelar' para que salga del mensaje y no se produzca ninguna acción.

Por ultimo voy a comentar dos partes del código que me parecen interesantes. La primera es una actividad que permite al usuario mandar mensajes desde la aplicación a la cuenta del twitter[13] del propio Club. Para ello he tenido que registrar previamente la aplicación en la página web de desarrolladores de Twitter y a partir de ahí con 4 claves que me dan poder permitir a la aplicación mandar mensajes desde esa cuenta de Twitter. Una de las principales utilidades de esta actividad es que el usuario puede mandar los resultados de los partidos de una manera cómoda, simplemente seleccionando un equipo del club y un club rival e introduciendo el resultado se crea un mensaje estándar que es enviado a twitter.

Para ello tengo una actividad que muestra la pantalla para enviar mensajes al twitter y al pulsar en el botón de enviar se ejecuta una tarea en fondo para enviar el mensaje. Además tengo una clase con los datos necesarios facilitados por twitter, CONSUMER_KEY, CONSUMER_SECRET, TOKEN y TOKEN_SECRET.

El último fragmento de código que me gustaría comentar es el encargado de escuchar si hay una llamada en el teléfono. El proceso que sigo es detectar que el estado del teléfono es RINGING (sonando) y defino la variable sonando como true y guardo el número de teléfono que llama. Lo siguiente seria detectar si es llamada perdida o recibida, puesto que a mí me interesan las llamadas perdidas si el usuario coge la llamada defino la variable callReceived como true, y si por el contrario la llamada no es atendida por defecto la variable callReceived está definida como false. Como el estado de llamada no atendida puede llevar a error, ya que el estado de reposo del teléfono se da en más de una situación, lo que hago es comprobar que las 2 variables mencionadas anteriormente son RING = TRUE y callReceived = false para ejecutar una actividad que me registra en la base de datos la llamada perdida y además me muestra una notificación. El código es el siguiente:

```
public class BroadcastReceiverTestActivity extends BroadcastReceiver {

static boolean ring=false;
static boolean callReceived=false;
private static String callerPhoneNumber;



@Override
public void onReceive(Context mContext, Intent intent)
{
       // Get the current Phone State
      String state = intent.getStringExtra(TelephonyManager.EXTRA_STATE);
      if(state==null)
          return;
      // If phone state "Rininging"
      if(state.equals(TelephonyManager.EXTRA_STATE_RINGING))
      {
                ring =true;
               // Get the Caller's Phone Number
               Bundle bundle = intent.getExtras();
               callerPhoneNumber= bundle.getString("incoming_number");
        }
      // If incoming call is received
      if(state.equals(TelephonyManager.EXTRA_STATE_OFFHOOK))
       {
              callReceived=true;
        }
       // If phone is Idle
      if (state.equals(TelephonyManager.EXTRA_STATE_IDLE))
      {
         if(ring==true&&callReceived==false)
            {
       Intent Lanzar = new Intent(mContext,
       AndroidNotificacionesActivity.class);
       Lanzar.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
       Lanzar.putExtra("telefono", callerPhoneNumber);
       mContext.startActivity(Lanzar);
       Toast.makeText(mContext, "It was A MISSED CALL from :
       "+callerPhoneNumber,
       Toast.LENGTH_LONG).show();
```

```
                }
    }}
```

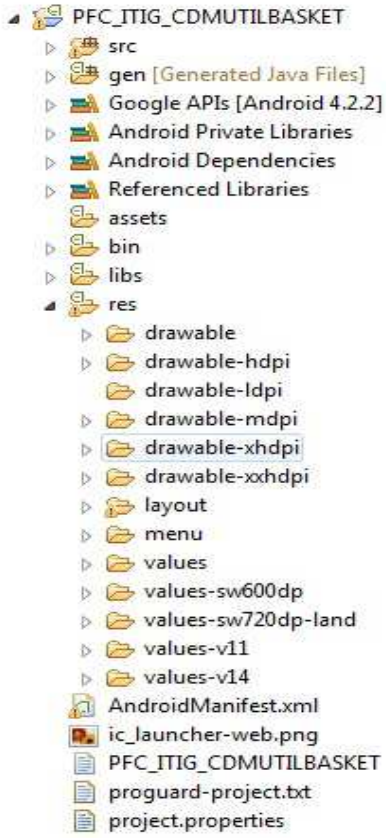
ESTRUCTURA DEL PROYECTO

Por último, con el fin de permitir el análisis del proyecto, su continuación o mejora, por parte de personas ajenas al desarrollo inicial se explicará la estructura que tiene, para así poder facilitar su entendimiento.

En primer lugar, se explicará cómo estructura Android los recursos necesarios para crear la interfaz de la aplicación. Para ello, al crear un proyecto se crea una carpeta llamada *res* que contiene varias carpetas dentro de ella. Las tres carpetas principales dentro de *res* son *drawable, layout* y *values.*

Drawable contiene las imágenes utilizadas por la aplicación, así como ficheros XML utilizados para crear el estilo de los botones y de las pestañas y poder mostrar una imagen cuando están pulsados y otra cuando no. Así pues, esta carpeta contiene las imágenes del proyecto y los ficheros XML utilizados para crear los estilos de las pestañas del apartado de consulta de datos.

Por su parte la carpeta *layout* contiene las vistas encargadas de crear la interfaz, es decir, los ficheros XML a partir de los cuales se crea la interfaz. Existen diversos tipos de carpetas *layout*, como por ejemplo *layout-hdpi, layout-medpi,* etc. Esta característica se utiliza para crear distintas interfaces según los distintos tipos de interfaz, tamaños u orientaciones. El proyecto hace uso de la carpeta *layout* para establecer la interfaz básica para dispositivos móviles con una resolución de tamaño igual a la del SAMSUNG GALAXY S5, puesto que que es el dispositivo móvil que tiene el usuario final.

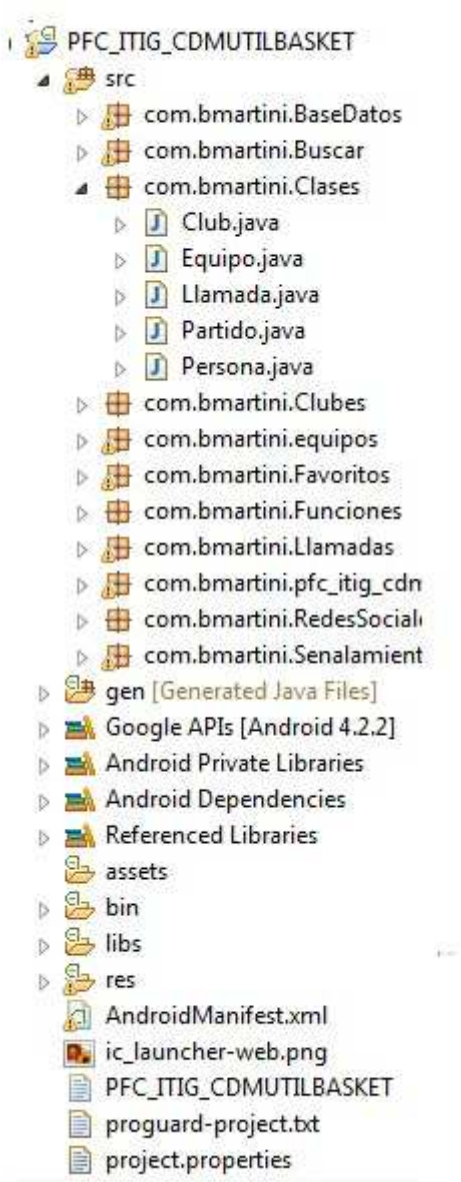


La última carpeta importante dentro de *res* es la carpeta *values*, utilizada para almacenar valores de la forma clave valor. Se utiliza principalmente para almacenar cadenas de caracteres y accederlas posteriormente haciendo uso del id. Es útil a la hora de realizar una aplicación puesto que permite modificar una cadena repetida varias veces a lo largo de la aplicación con solo cambiarla en un fichero. Dentro de *values* encontramos los archivos *string* y *arrays*. El primero de ello se usa para almacenar simples cadenas y el segundo permite crear arrays de cadenas, utilizados en los *spinners*, también llamados menús desplegables.

Ademas dentro de values encontramos un archivo colors.xml, en el que he introducido los códigos de una amplia gama de colores puesto que es una tarea muy laboriosa tener que meter el codiogo del color cada vez que lo necesitemos.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="grey_tabla">#7E7974</color>
    <color name="white">#FFFFFF</color>
    <color name="yellow">#FFFF00</color>
```

En cuanto al código que implementa las funcionalidades puede encontrarse en la carpeta *src.* Aquí hay que hacer una distinción entre archivos que actúan como controladores y archivos que actúan como objetos. Los primeros forman parte del patrón modelo-vista-controlador y se utilizan para comunicar las interacciones del usuario con el dispositivo con los cambios que se requieren hacer en el modelo. Los segundos se utilizan para crear los objetos utilizados en la funcionalidad y se encuentran en la carpeta *classes.* Además he separado cada apartado del menú para tener más claridad a la hora de trabajar con el proyecto.



6. EVALUACIÓN Y PRUEBAS

Para comprobar el correcto funcionamiento de la aplicación se han llevado a cabo diversas evaluaciones y pruebas descritas a continuación.

6.1. EVALUACIÓN

Una vez terminada la implementación se han realizado distintas pruebas para garantizar el correcto funcionamiento, probando cada funcionalidad una a una y corrigiendo aquellas en las que se presentaba algún error, de forma que se ha conseguido depurar la aplicación lo máximo posible con el fin de evitar fallos a la hora de hacer uso del proyecto desarrollado.

6.2. PRUEBAS

Una de las pruebas más importantes era probar la aplicación en funcionamiento en el dispositivo, para ello se ha empleado el simulador incorporado en el propio SDK de Android. Mediante el simulador se pueden configurar diversos dispositivos virtuales, de forma que simulen las capacidades técnicas de un dispositivo real.

Además posteriormente se ha empleado el mismo dispositivo móvil al que va destinado esta aplicación, SAMSUNG GALAXY S5 con la versión más reciente de Android, Lollipop para verificar su correcto funcionamiento.

A continuación se muestran distintas imágenes de la aplicación desarrollada funcionando en el SAMSUNG GALAXY S5.


64% 20:06
CDMUTILBASKET
MB
MUTILBASKET
VALLE DE ARANGUREN - NAVARRA


64% 20:07
CDMUTILBASKET


63% 20:08
CDMUTILBASKET
NOMBRE
PRUEBA
APELLIDOS
PRUEBA
EQUIPO
Empleados
CLUB
C.D. MUTILBASKET
FECHA
TELEFONO
948163595
EMAIL
MUTILBASKET
VALLE DE ARANGUREN - NAVARRA


64% 20:07
CDMUTILBASKET
FEDERACION NAVARRA BALONCESTO
basket navarra CLUB
FEB
MB
ACB
MUTILBASKET
VALLE DE ARANGUREN - NAVARRA

7. CONCLUSIONES

7.1 CONCLUSIONES

El último apartado a tratar es el relacionado con las conclusiones del proyecto que se ha desarrollado. Cabe mencionar que el proyecto, a lo largo de todo su desarrollo, ha logrado cumplir con los objetivos y motivaciones que se habían marcado al iniciar el proyecto.

He podido realizar el desarrollo de una aplicación destinada a dispositivos móviles desde cero hasta lograr una aplicación que funciona de la manera esperada. De esta forma, he adquirido conocimientos relacionados con el desarrollo de aplicaciones para dispositivos móviles, en este caso la programación destinada a Android así como el patrón modelo-vista-controlador. Además, se han empleado conocimientos ya aprendidos como el uso de base de datos mediante el sistema de gestión SQLite. Por otro lado, el hecho de tener que desarrollar la aplicación de forma individual, sin un código inicial o referencia ha hecho que consultar diferentes fuentes tales como páginas web haya cobrado gran relevancia a la hora de abordar y solventar diversos problemas o cuestiones surgidos a lo largo del desarrollo.

Una de las etapas más costosas a la hora de desarrollar el proyecto fueron el inicio del desarrollo y la creación de las interfaces. El inicio del desarrollo fue complicado debido a la diferencia existente entre el desarrollo para Android del desarrollo de una aplicación de escritorio usando Java. Esto supuso tener que realizar una adaptación al método de desarrollo para Android con sus peculiaridades, a pesar de esto, al tener conocimientos del lenguaje Java no conllevo graves problemas o estancamientos a la hora de desarrollar, pero sí algo de lentitud a la hora de iniciar el desarrollo. Por otro lado, la creación de las interfaces es dificultosa debido a que la herramienta empleada no está debidamente desarrollada por lo que es costoso ajustar los distintos componentes de la interfaz. Cabe también mencionar la poca optimización a la hora de emplear un dispositivo virtual, lo que lleva a que las pruebas sean un proceso lento debido a la poca fluidez que presenta esta característica del SDK.

Como conclusión mencionar que el proyecto me ha servido para aprender el uso de tecnologías antes desconocidas como es la programación para dispositivos Android, así como para poner en práctica conocimientos adquiridos durante la carrera, como por ejemplo los referentes al uso y gestión de la base de datos.

7.1 LINEAS FUTURAS

La aplicación está destinada a un único usuario. EL trabajo futuro será hacer que la aplicación pueda usarse por cualquier club deportivo. Para ello algunas tareas para el futuro serán:

- Crear una forma de cargar los datos a la aplicación de una manera más cómoda para el usuario que desde la aplicación, vía web o desde un fichero externo a la aplicación.
- Mantener la aplicación actualizada. Android es un sistema operativo en constante evolución, y por lo tanto la aplicación debería ajustarse de la mejor manera posible al ritmo del sistema operativo.
- La interfaz, es uno de los talones de Aquiles de la aplicación, puesto que requiere un poco más de experiencia en diseño de interfaces.
- Permitir a la aplicación conectarse a cualquier cuenta de twitter sin tener que hacerlo a través del código.

8. BIBLIOGRAFIA

- Páginas web
    - 1. Android Developers (http://developer.android.com/index.html)
    - 2. Sgoliver (http://www.sgoliver.net/blog/curso-de-programacion-android/)
    - 3. Wikipedia, the free encyclopedia (http://en.wikipedia.org/)
    - 4. Eclipse – The Eclipse Foundation (http://www.eclipse.org/)
    - 5. Androideity (http://androideity.com/)
    - 6. Stack Overflow (http://stackoverflow.com/)
    - 7. Android (https://www.android.com/)
    - 8. JAVA (https://www.java.com/es/)
    - 9. SQLite (https://www.sqlite.org/)
    - 10. C.D. MUTILBASKET (www.mutilbasket.com)
    - 11. SAMSUNG (www.samsung.com/)
    - 12. UML (www.uml.org/)
    - 13. TWITTER (https://dev.twitter.com/)

APLICACIÓN ANDROID PARA GESTIÓN DE UN CLUB DEPORTIVO

Borja Martinicorena Huguet

ÍNDICE

INTRODUCCIÓN

- Aplicación Android:
  - Adquisición Google
  - Cuota de mercado
  - Dispositivos
  - Versiones
  - Lenguaje de programación JAVA
  - Almacenamiento SQLite

INTRODUCCIÓN

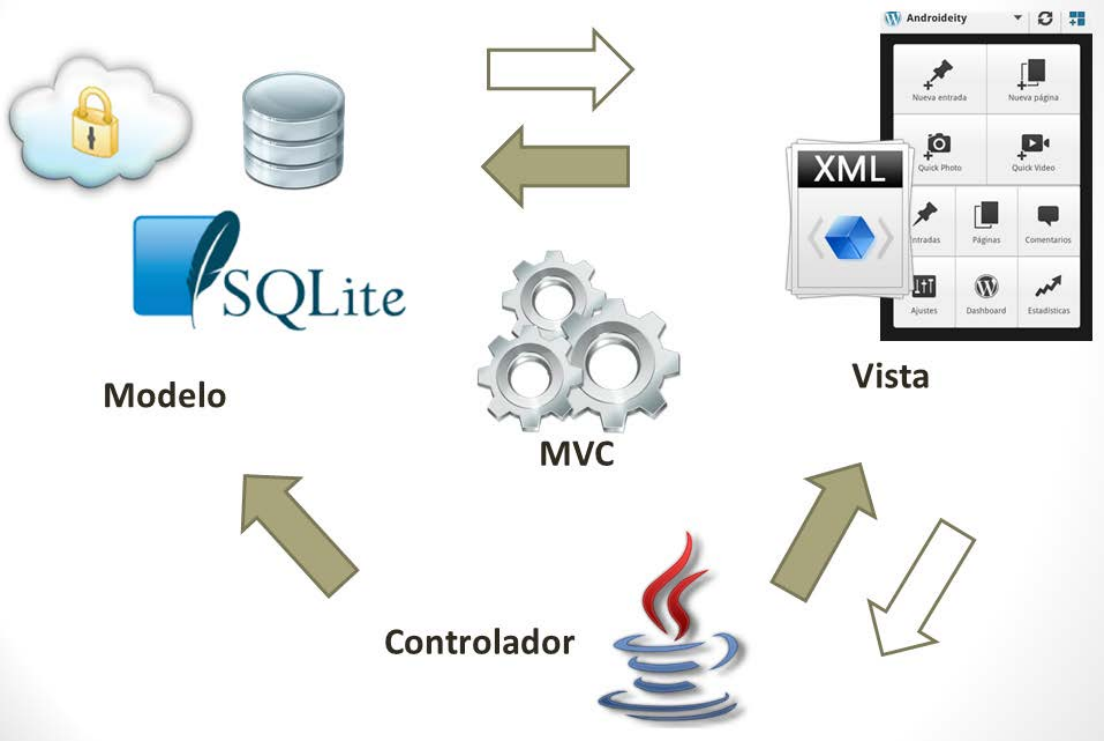
- Elección del proyecto
- Primeros pasos
- Trabajo con el cliente

MUTILBASKET
VALLE DE ARANGUREN - NAVARRA

HERRAMIENTAS UTILIZADAS

- Ordenador con Sistema Operativo Windows 7
- Entorno de desarrollo :
  - ECLIPSE
  - SDK de Android
- Smartphone Samsung Galaxy S5

PATRÓN MVC: Modelo Vista Controlador



PATRÓN MVC: Modelo Vista Controlador

MODELO:

- Representaciones que construiremos basadas en la información con la que operará nuestra aplicación
- SQLite:
  - Modelo para almacenamiento
  - Manejador Open Source
  - Trabaja con poca memoria
  - Velocidad aceptable
  - Integrada dentro de la aplicación
  - Creación a través de código

PATRÓN MVC: Modelo Vista Controlador

VISTA:

- Interfaz con la que interactúa el usuario
- XML (Especificación para diseñar lenguajes de marcado, que permite definir etiquetas personalizadas para descripción y organización de datos)
- Photoshop
- Tamaños de pantalla

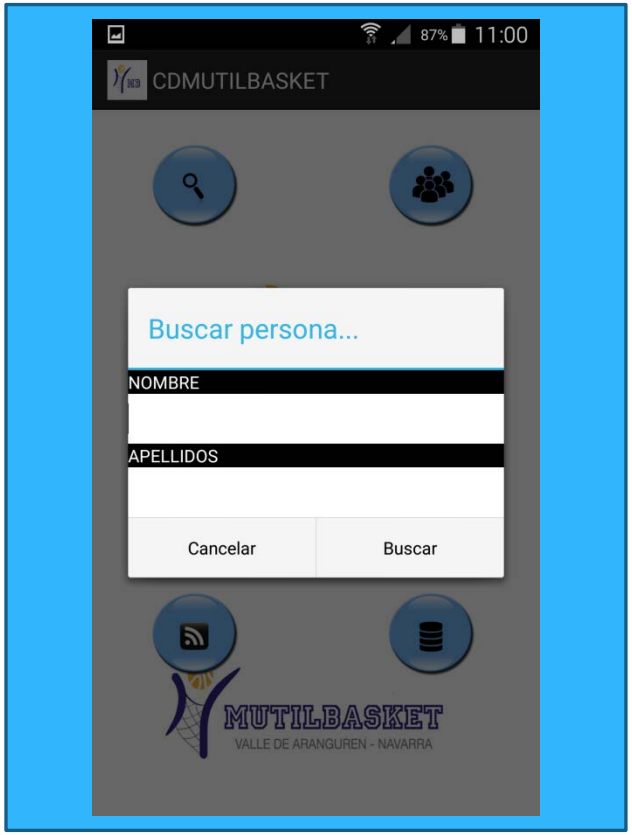
PATRÓN MVC: Modelo Vista Controlador

CONTROLADOR:

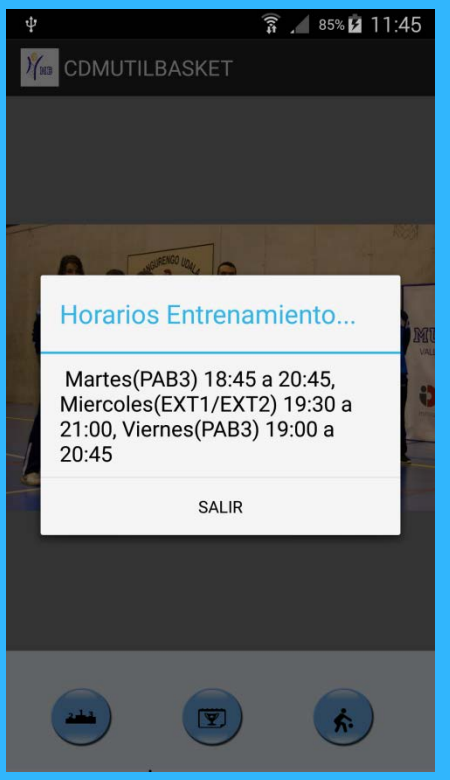
- Clases que nos ayudaran a dar funcionalidad a nuestras vistas.
- Están programados en lenguaje JAVA
- Uso de Activity

FUNCIONALIDAD DE LA APLICACIÓN

FUNCIONALIDAD DE LA APLICACIÓN

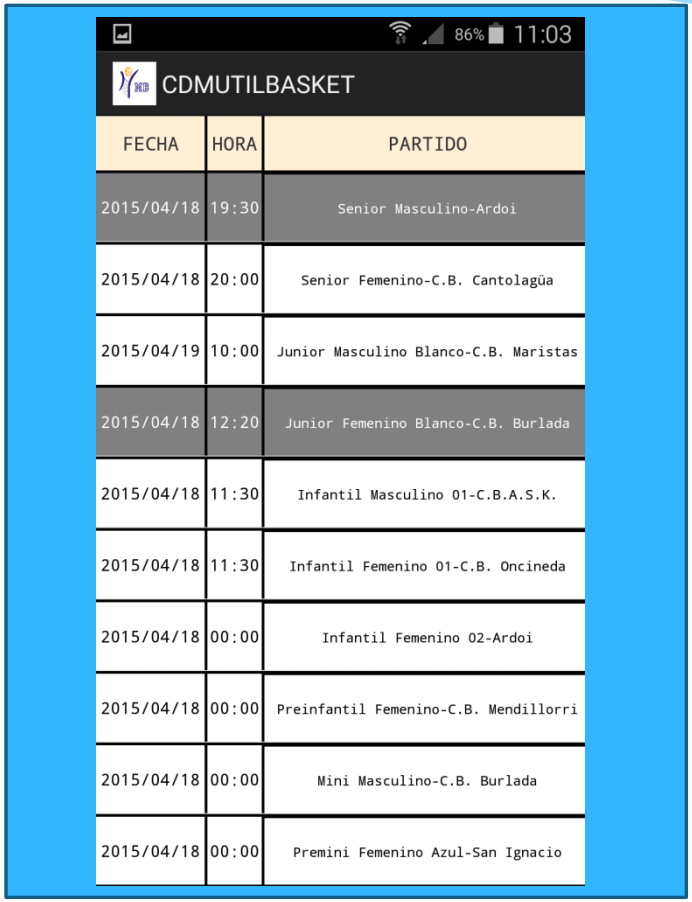


FUNCIONALIDAD DE LA APLICACIÓN



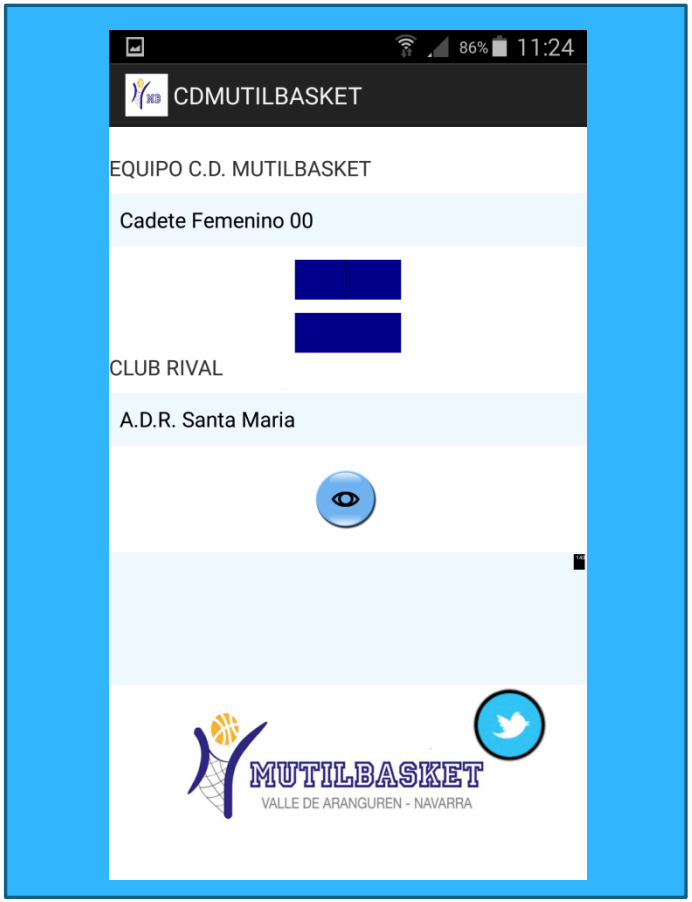
FECHA	HORA	RIVAL
2014/10/05	11:30	C.D.MUTILBASKET
2014/10/11	12:10	Lagunak
2014/10/18	10:50	San Ignacio
2014/10/25	09:30	San Cernin
2014/10/31	19:15	C.B. Burlada
2014/11/08	00:00	Navarro Villoslada
2014/11/22	00:00	Liceo Monjardin
2014/11/29	00:00	C.B. Mendillorri
2014/12/13	00:00	Larraona
2014/12/20	00:00	Ardoi
2015/01/10	00:00	C.B. Oncineda
2015/01/17	00:00	C.D. Teresianas
2015/01/24	11:30	C.D.MUTILBASKET

FUNCIONALIDAD DE LA APLICACIÓN

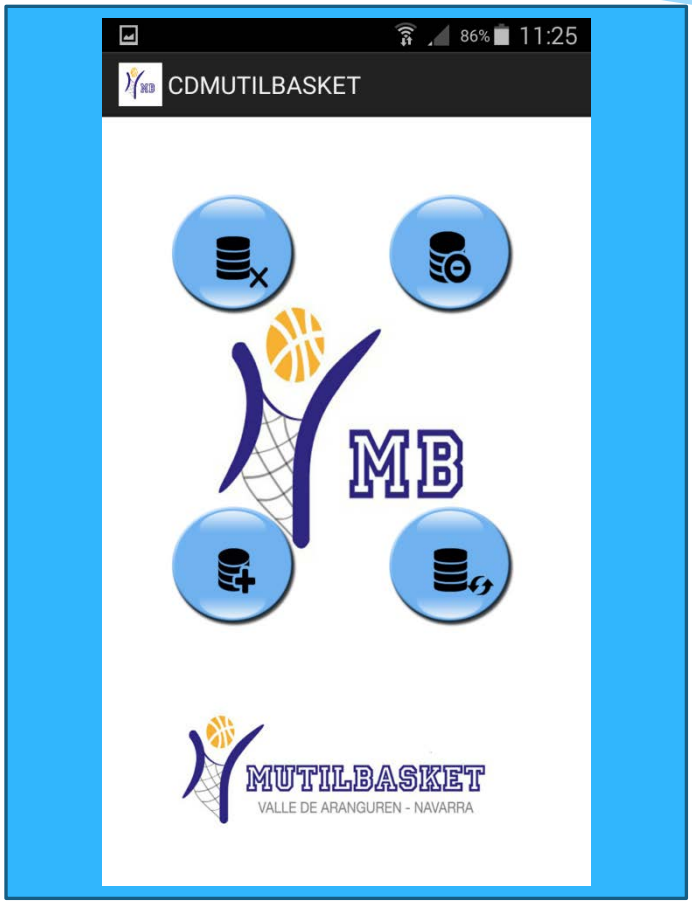


FUNCIONALIDAD DE LA APLICACIÓN

FUNCIONALIDAD DE LA APLICACIÓN



FUNCIONALIDAD DE LA APLICACIÓN



CONCLUSIONES

- Utilidad
- Android
- Líneas futuras